# Chapter 7

# More SQL: Complex Queries, Triggers, Views, and Schema Modification

## Outline of the chapter:

* More Complex SQL Retrieval Queries
* Specifying Semantic Constraints as Assertions and Actions as Triggers
* Views (Virtual Tables) in SQL
* Schema Modification in SQL

## Comparisons Involving NULL

* What is a NULL?
* Unknown or unavailable value
* NULL is not a data value! We cannot use equal operator = to check null attributes!
* SQL allows queries that check whether an attribute value is NULL by IS NULL or IS NOT NULL.
* **Query 18.** Retrieve the names of all employees who do not have supervisors.

SELECT fname, lname

FROM employee

WHERE employee.super\_ssn IS NULL;

## Nested Queries

* What is a Nested Query?
  1. Using select-from-where statement within WHERE clause of another query.
* Example:

SELECT \*

FROM employee

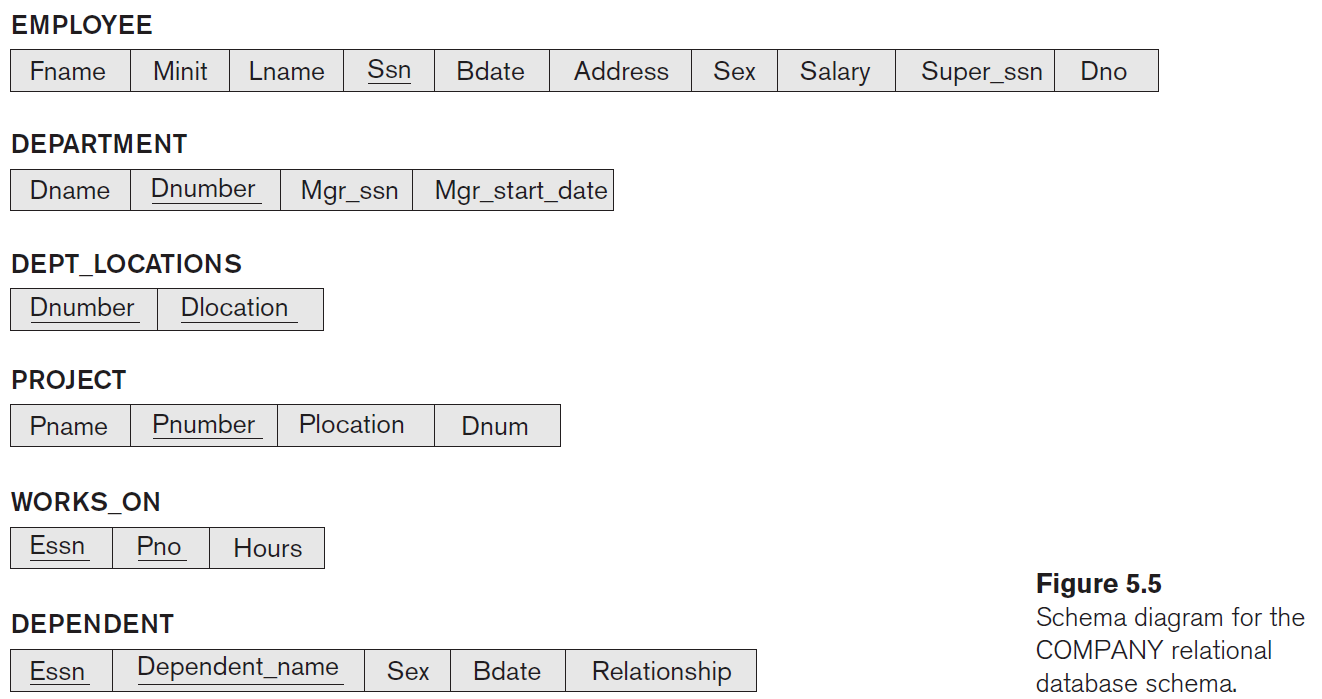
WHERE dno IN (SELECT dnumber

FROM department

WHERE dname='Research');

* They are also called Correlated Nested Queries.
* To understand nested queries, consider them as two for-loops inside each other;
  1. For each tuple in outer query, run the inner query and see if there is any result.

To write queries you may need to look at the schema diagram of COMPANY database:



Write a query which prints first name and last name of all department managers (write it as nested queries and with IN operator).

* Can you write a query which only returns managers’ SSN?
* In which table are first name and last name?
* How could you know if an employee is a department manager?
* Now compile everything as a nested query:

What the following query will return?

SELECT DISTINCT pnumber

FROM project

where pnumber IN

(SELECT pnumber

FROM project, department, employee

where project.dnum= department.dnumber and

department.mgr\_ssn = employee.ssn and employee.lname = 'Smith')

OR

pnumber IN

(SELECT pno

FROM works\_on, employee

WHERE works\_on.essn = employee.ssn and

employee.lname = 'Smith');

Use a tuple of values in comparisons, by placing them within parentheses.

SELECT DISTINCT essn

FROM works\_on

where (pno, hours) IN (SELECT pno, hours

FROM works\_on

where essn = '123456789');

* What does the query return?

### Comparing a value to a set/multiset

Operators:

* 1. IN
  2. = ANY (or = SOME) operator
     1. v = ANY V
     2. Returns TRUE if the value *v* is equal to some value in the set *V*.
     3. It is equivalent to IN.
  3. Operators >, >=, <, <=, and <> could also be combined with ANY (or SOME).
  4. ANY could be replaced with ALL: meaning condition must be true for all values in the set.

Example:

SELECT lname, fname

FROM employee

where salary >ALL (SELECT salary

FROM employee

where dno = 5);

## EXISTS function

* + Check whether the result of a nested query is empty or not.
  + It is a Boolean function that returns a TRUE or FALSE.
  + Could be used with NOT; NOT EXISTS

Example:

SELECT fname, lname

FROM employee

WHERE not EXISTS **(SELECT \***

**FROM dependent**

**WHERE employee.ssn = dependent.essn)**;

What does the query return?

How about this?

SELECT fname, lname

FROM employee

WHERE EXISTS (SELECT \*

FROM dependent

WHERE employee.ssn = dependent.essn)

AND

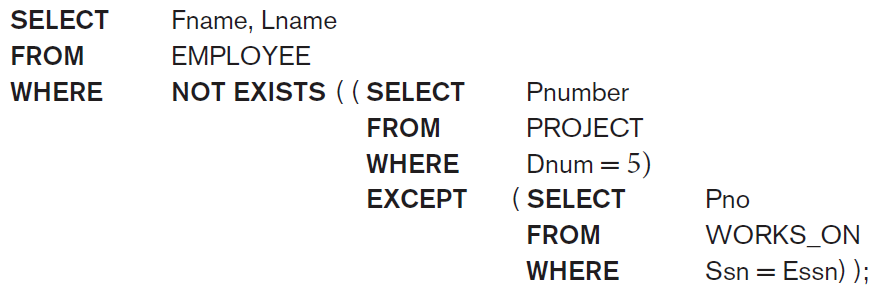
EXISTS (SELECT \*

FROM department

WHERE employee.ssn = department.mgr\_ssn);

## A challenging example

Query: List first and last name of employees who work on ALL projects controlled by Dno=5.



EXCEPT operator is not implemented in mysql, there is a lab question asking you to rewrite this query such that you could run it in mysql. Hint: instead of EXCEPT use NOT IN, but you need to change the query a bit to make it work.

## Explicit Sets

* You can use explicit set of values in WHERE clause.

SELECT DISTINCT essn

FROM works\_on

where pno IN (1, 2, 3);

## Renaming of Attributes in a query

* Use qualifier AS followed by a new name, to rename an attribute that appears in the result of a query.

SELECT E.lname AS Employee\_name, S.lname as Supervisor\_name

FROM employee as E, employee as S

WHERE E.super\_ssn = S.ssn;

## Joining Tables

* It allows you to join tables in the FROM clause.
* It is easier to comprehend, than specifying the table connections within WHERE clause.
* What this query retrieve?

SELECT fname, lname, address

FROM employee, department

where employee.dno = department.dnumber and department.dname = 'Research';

* Using join operator:

SELECT fname, lname, address

FROM (employee JOIN department ON employee.dno = department.dnumber)

where department.dname = 'Research';

### Different types of join

* INNER JOIN
  + Default type of join in a joined table.
  + Tuple is included in the result only if a matching tuple exists in the other relation.
  + The join condition is specified explicitly. (As example Dno = Dnumber).
* NATURAL JOIN on two relations R and S
  + No join condition is specified.
  + R an S are joined on attribute(s) with the same name.
  + You may need to rename attributes of one relation so it can be joined with another using NATURAL JOIN.
  + Example:

SELECT fname, lname, address

FROM (employee NATURAL JOIN

(department AS DEPT (dname, dno, mssn, msdate)))

WHERE dname='Research';

The implicit join condition is EMPLOYEE.Dno = DEPT.Dno.

*We cannot run the above query, because renaming entire table like ‘department AS DEPT (Dname, Dno, Mssn, Msdate)’ is not supported in mysql.*

* OUTER Joins
* LEFT OUTER JOIN
  1. Every tuple in left table must appear in result.
  2. If there is no matching tuple on right table, pad it with NULL values for attributes of right table.

Example:

Q8B:

SELECT E.lname AS Employee\_name, S.lname as Supervisor\_name

FROM (employee as E LEFT OUTER JOIN employee as S ON E.super\_ssn = S.ssn);

* RIGHT OUTER JOIN
  1. Every tuple in right table must appear in result.
  2. If no matching tuple on left table, pad it with NULL values for attributes of left table.